Journal of Computational Mathematics, Vol.9, No.1, 1991, 33–40.

A PARALLEL ALGORITHM FOR TOEPLITZ TRIANGULAR MATRICES*[1)]

Chen Ming-kui Lu Hao
(*Department of Mathematics, Xi'an Jiaotong University, Xi'an, China*)

Abstract

A new parallel algorithm for inverting Toeplitz triangular matrices as well as solving Toeplitz triangular linear systems is presented in this paper. The algorithm possesses very good parallelism, which can easily be adjusted to match the natural hardware parallelism of the computer systems, that was assumed to be much smaller than the order n of the matrices to be considered since this is the usual case in practical applications. The parallel time complexity of the algorithm is $O([n/p]\log n + \log^2 p)$, where p is the hardware parallelism.

§1. Introduction

Parallelly inverting triangular matrices and solving triangular linear systems is an interesting problem both in theory and practice. The best parallel algorithm known so far requires $O(\log^2 n)$ time steps and $O(n^3)$ processors, where n is the order of the matrices[5,6]. The order of the time complexity can not be reduced further even for more strongly structured triangular matrices, but the number of processors needed can be reduced. Although the approximate algorithm for parallelly solving Toeplitz triangular systems presented in [1] reduces the time complexity, it requires precomputations, and does not seem practical since some restriction must be imposed on the parameter ε to ensure the nonsingularity of matrix A_ε[1]. Chen and Lu[2] constructed an algorithm for inverting Toeplitz triangular matrices and solving Toeplitz triangular linear systems, by which the number of processors needed to perform the algorithm can be reduced to n.

In practical applications, the number of processors of a computer system, denoted by p, is limited and frequently much smaller than n, the order of the matrices. We will consider the problem on parallelly inverting Toeplitz triangular matrices as well as solving the associated linear systems in this case. The parallel time complexity of the algorithm presented here is $O([n/p]\log n + \log^2 p)$, where $\log n$ means $\log_2 n$ and $[x]$ is the integer ceiling function of x.

We will first give a method to carry out multiplication of a vector by a circulant or a block circulant matrix in §2, and then develop an algorithm for computing the product of the Toeplitz or the block Toeplitz matrix and vector in §3. In §4, the method for inverting Toeplitz triangular matrices as well as solving the associated Toeplitz systems will be constructed.

---

* Received April 2, 1987.
1) This work was supported by the National Natural Science Foundation of China.

§2. Circulant Matrices and Block Circulant Matrices

Consider the following special class of Toeplitz matrices

$$C = \begin{bmatrix} c_0 & c_1 & c_2 & \cdots & c_{q-1} \\ c_{q-1} & c_0 & c_1 & \cdots & c_{q-2} \\ c_{q-2} & c_{q-1} & c_0 & \cdots & c_{q-3} \\ \vdots & \vdots & \vdots & & \vdots \\ c_1 & c_2 & c_3 & \cdots & c_0 \end{bmatrix}, \tag{2.1}$$

which are called circulant matrices. This kind of matrices are completely defined by their first row, and thus frequently denoted by

$$C = \text{ circ } (c_0, c_1, c_2, \cdots, c_{q-1}).$$

Circulant matrices can be diagonalized by the Fourier matrix $F = (f_{ij})_{q\times q}$ with elements $f_{ij} = q^{-1/2}\omega^{-(i-1)(j-1)}$ $(i, j = 1, 2, \cdots, q)$, where ω is the primitive nth root of unity[4], i.e., it holds for any circulant matrices that

$$C = F^H D F, \tag{2.2}$$

where

$$D = \text{ diag } (\lambda_0, \lambda_1, \cdots \lambda_{q-1}), \tag{2.3}$$

and the eigenvalues λ_k's of C are defined by

$$\lambda_k = \sum_{l=0}^{q-1} c_l \omega^{kl}, \quad k = 0, 1, \cdots, q-1. \tag{2.4}$$

It is easy to see that premultiplying a vector by matrices F and F^H may be accomplished by Fast Fourier Transform (FFT) and its inverse, respectively, and the eigenvalues λ_k's can be computed via FFT[4]. Thus, multiplying any q-vector by a circulant can be accomplished in $(3\log q + 1)$ time steps with q processors[2].

A block matrix of the form

$$C_b = \text{ circ } (T_0, T_1, T_2, \cdots, T_{q-1}),$$

where each of the blocks T_j is a pth order matrix, is called a block circulant matrix. It is easily verified that

$$C_b = \sum_{k=0}^{q-1} P^k \otimes T_k, \tag{2.5}$$

where the notation $\otimes$ denotes the Kronecker product of matrices, and

$$P = \text{ circ } (0, 1, 0, \cdots, 0)$$

is a circulant of order q, and that (see [4])

$$P = F^H \tilde{D} F, \tag{2.6}$$

where

$$\tilde{D} = \text{ diag } (1, \omega, \omega^2, \cdots, \omega^{q-1}).$$

We will now derive an algorithm for multiplying a (pq)-vector d by the block circulant matrix C_b. Partitioning vector d as

$$d = \begin{bmatrix} d^{(0)} \\ d^{(1)} \\ \vdots \\ d^{(q-1)} \end{bmatrix}, \tag{2.7}$$

where each of $d^{(j)}$'s is a vector of length p, we can express the product of C_b and d as

$$C_b d = \Big\{ \sum_{k=0}^{q-1} P^k \otimes T_k \Big\} (F \otimes I_p)^{-1} (F \otimes I_p) d = \Big\{ \sum_{k=0}^{q-1} (P^k \otimes T_k)(F^H \otimes I_p) \Big\} (F \otimes I_p) d, \tag{2.8}$$

where I_p is the pth order identity matrix.

If the (pq)-vector

$$y = (F \otimes I_p) d \tag{2.9}$$

is similarly partitioned into q subvectors $y^{(0)}, y^{(1)}, \cdots, y^{(q-1)}$ as in (2.7), then the $y^{(j)}$'s are given by

$$y^{(j)} = q^{-1/2} \sum_{l=0}^{p-1} \omega^{-jl} d^{(l)}, \quad j = 0, 1, \cdots, q-1. \tag{2.10}$$

From (2.6) it can be derived that

$$\sum_{k=0}^{q-1} (P^k \otimes T_k)(F^H \otimes I_p) = (F^H \otimes I_p) \sum_{k=0}^{q-1} (\tilde{D}^k \otimes T_k). \tag{2.11}$$

Substituting it into (2.8) we have

$$u = C_b d = (F^H \otimes I_p) z, \tag{2.12}$$

where

$$z = Xy, \tag{2.13}$$

and

$$X = \sum_{k=0}^{q-1} (\tilde{D}^k \otimes T_k) = \text{ diag } (X_0, X_1, \cdots, X_{q-1}) \tag{2.14}$$

is a (pq)th order block diagonal matrix with the matrices

$$X_i = \sum_{k=0}^{q-1} \omega^{ik} T_k, \quad i = 0, 1, \cdots, q-1 \tag{2.15}$$

as its diagonal blocks.

If the vector z is partitioned into subvectors $z^{(j)} (j = 0, 1, \cdots, q-1)$ in the same way as in (2.7), then the $z^{(j)}$'s are defined by

$$z^{(j)} = X_j y^{(j)}, \quad j = 0, 1, \cdots, q-1. \tag{2.16}$$

Similarly, equation (2.12) can be written as

$$u^{(j)} = q^{-1/2} \sum_{l=0}^{p-1} \omega^{jl} z^{(l)}, \quad j = 0, 1, \cdots, q-1. \tag{2.17}$$

Thus, multiplication of a block circulant matrix and a vector can be computed via (2.10), (2.15), (2.16), and (2.17), each of which, except (2.16), that expresses the product of a matrix and a vector, may simultaneously be accomplished by using FFT's or their inverses, and, therefore, is suitable to compute in parallel.

§3. Toeplitz and Block Toeplitz Matrices

In this section, we will use the method developed in the previous section to construct a parallel algorithm to compute the product of a Toeplitz or block Toeplitz matrix and a vector.

Let $T = (t_{ij})$ be a Toeplitz matrix of order q defined by $t_{ij} = t_{j-i}$, i.e.,

$$T = \begin{bmatrix} t_0 & t_1 & \cdots & t_{q-1} \\ t_{-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_1 \\ t_{1-q} & \cdots & t_{-1} & t_0 \end{bmatrix}. \tag{3.1}$$

If we let

$$\tilde{T} = \begin{bmatrix} 0 & t_{1-q} & \cdots & t_{-1} \\ t_{q-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & t_{1-q} \\ t_1 & \cdots & t_{q-1} & 0 \end{bmatrix} \tag{3.2}$$

be a Toeplitz matrix of order q, then

$$C = \begin{bmatrix} T & \tilde{T} \\ \tilde{T} & T \end{bmatrix} = \text{circ}\,(t_0, t_1, \cdots, t_{q-1} 0, t_{1-q}, \cdots, t_{-1}) \tag{3.3}$$

is a circulant of order $2q$. The product

$$C \begin{bmatrix} b \\ 0 \end{bmatrix} = \begin{bmatrix} Tb \\ \tilde{T}b \end{bmatrix}, \tag{3.4}$$

which shows that the first q components of the product constitute the Toeplitz matrix-vector product Tb, can be computed in $6 \log q + 8$ time steps with q processors, and so can the product Tb.

The method developed above can easily be generalized to the block Toeplitz matrix. Let

$$T_b = \begin{bmatrix} T_0 & T_1 & \cdots & T_{q-1} \\ T_{-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & T_1 \\ T_{1-q} & \cdots & T_{-1} & T_0 \end{bmatrix} \tag{3.1'}$$

be a block Toeplitz matrix with blocks T_j of order p. Similarly, let

$$\tilde{T}_b = \begin{bmatrix} 0 & T_{1-q} & \cdots & T_{-1} \\ T_{q-1} & \ddots & \ddots & \vdots \\ \vdots & \ddots & \ddots & T_{1-q} \\ T_1 & \cdots & T_{q-1} & 0 \end{bmatrix} \tag{3.2'}$$

be a block Toeplitz matrix. Then

$$C_b = \begin{bmatrix} T_b & \tilde{T}_b \\ \tilde{T}_b & T_b \end{bmatrix} \tag{3.3'}$$

is a block circulant matrix of order $2pq$, and the product of T_b and a vector d is the (pq)-vector consisting of the first pq components of $C_b \begin{bmatrix} d \\ 0 \end{bmatrix}$, and therefore, can be computed by using the method described in Section 2.

It is more suitable for our purpose to assume that the blocks T_j's are Toeplitz matrices. It follows that the pth order matrix X_i's defined by equation (2.15) are Toeplitz matrices, and the product defined in (2.16) can be carried out in $2q(6\log p+8)$ steps with parallelism p. As we pointed out above, the computations in (2.10), (2.15), and (2.17) can be accomplished by performing FFT of length q in $2q(\log q+1)$, $4q(\log q+1)$ and $2q(\log q+1)$ steps, respectively. The procedure to compute the block Toeplitz matrix vector product is summarized in Table 1.

Table 1. Multiplication of a vector by a block Toeplitz matrix

stage	formula	time steps	parallelism
1	(2.10)	$2q(\log q + 1)$	p
2	(2.15)	$4q(\log q + 1)$	p
3	(2.16)	$2q(6\log p + 8)$	p
4	(2.17)	$2q(\log q + 1)$	p

The overall time steps needed are $8q(\log q + 3) + 12q\log p$, where $n = pq$ is the order of the block Toeplitz matrix to be used to multiply a vector.

§4. Toeplitz Triangular Matrices

We now consider the problem on parallelly inverting Toeplitz triangular matrices and solving linear systems having a Toeplitz triangular coefficient matrix. We have constructed an algorithm in [2] to obtain the inversion of a Toeplitz triangular matrix of order n in $(3\log^2 n + 5\log n)$ steps on a computer system having n processors. The parallel algorithm for the same problem on a computer with less processors will be developed in this section.

Assume for convenience that there are p processors available in the computer system, and that $n = pq, q = 2^r$ for some positive integer r without loss of generality. Let

$$U = \begin{bmatrix} t_0 & t_1 & \cdots & t_{n-1} \\ & \ddots & \ddots & \vdots \\ & & \ddots & t_1 \\ & & & t_0 \end{bmatrix}$$

be a Toeplitz upper triangular matrix. It is well known[7] that the inverse of U is a Toeplitz upper triangular matrix, and completely defined by its last column, which is the solution of the following linear system

$$Uu = e_n \tag{4.1}$$

where e_n is the last column of the nth order identity matrix.

To construct the method for solving (4.1), we partition matrix U into pth order blocks as follows

$$U = \begin{bmatrix} U_0 & U_1 & \cdots & U_{q-1} \\ & \ddots & \ddots & \vdots \\ & & \ddots & U_1 \\ & & & U_0 \end{bmatrix}, \tag{4.2}$$

where U_0 is a Toeplitz upper triangular matrix and $U_j (j = 1, 2, \cdots, q-1)$ are Toeplitz matrices. To conform with the block structure of the matrix U, we partition the unknown vector u into q p-vectors as

$$u = \begin{bmatrix} u_{q-1} \\ u_{q-2} \\ \vdots \\ u_0 \end{bmatrix}. \tag{4.3}$$

Now let $u^{(k)}$ denote the $p2^k$-subvector of u consisting of the last $p2^k$ components of u, i.e.

$$u^{(k)} = \begin{bmatrix} u_{2^k-1} \\ \vdots \\ u_1 \\ u_0 \end{bmatrix} \tag{4.4}$$

which satisfies

$$U^{(k)} u^{(k)} = (0, 0, \cdots, 0, 1)^T, \quad k = 1, 2, \cdots, r, \tag{4.5}$$

where

$$U^{(k)} = \begin{bmatrix} U_0 & U_1 & \cdots & U_{2^k-1} \\ & \ddots & \ddots & \vdots \\ & & \ddots & U_1 \\ & & & U_0 \end{bmatrix}$$

is the submatrix of order $p2^k$ at the lower right corner of U. By partitioning $U^{(k)}$ as

$$U^{(k)} = \begin{bmatrix} U_{11}^{(k)} & U_{12}^{(k)} \\ & U_{11}^{(k)} \end{bmatrix}, \tag{4.6}$$

where

$$U_{11}^{(k)} = \begin{bmatrix} U_0 & U_1 & \cdots & U_{2^{k-1}-1} \\ & \ddots & \ddots & \vdots \\ & & \ddots & U_1 \\ & & & U_0 \end{bmatrix} \tag{4.7}$$

is a Toeplitz upper triangular matrix, and

$$U_{12}^{(k)} = \begin{bmatrix} U_{2^{k-1}} & \cdots & \cdots & U_{2^k-1} - 1 \\ \vdots & \ddots & & \vdots \\ \vdots & & \ddots & \vdots \\ U_1 & \cdots & \cdots & U_{2^k} \end{bmatrix}, \tag{4.8}$$

and noting that $(U_{11}^{(k)})^{-1}$ is completely defined by $u^{(k)}$, the system

$$u_1^{(k)} = -(U_{11}^{(k)})^{-1} U_{12}^{(k)} u_2^{(k)}, \tag{4.9}$$

$$u_2^{(k)} = u^{(k-1)}. \tag{4.10}$$

where

$$u_1^{(k)} = \begin{bmatrix} u_{2^k-1} \\ \vdots \\ u_{2^{k-1}} \end{bmatrix}, \tag{4.11}$$

$$u^{(k)} = \begin{bmatrix} u_1^{(k)} \\ u_2^{(k)} \end{bmatrix}, \tag{4.12}$$

which is equivalent to (4.5), can be recursively solved if the inverse U_0^{-1} has been computed, and finally, we obtain the desired vector u, which is given by $u^{(r)}$. The algorithm proceeds as follows.

Algorithm PITM (Parallel Inversion of Toeplitz Triangular Matrices).

1. Compute the inverse of U_0 to get u_0 (see [2] Algorithm II) and compute

$$u_1 = -U_0^{-1} U_1 u_0,$$

and then form

$$u^{(1)} = \begin{bmatrix} u_1 \\ u_0 \end{bmatrix}.$$

2. For $k = 2, 3, \cdots, r$ do
   1) Let $u_2^{(k)} = u^{(k-1)}$.
   2) Calculate $\tilde{u}_1^{(k)} = -U_{12}^{(k)} u_2^{(k)}$.

3) Use $u^{(k-1)}$ to form $(U_{11}^{(k)})^{-1}$ and then multiply $\tilde{u}_1^{(k)}$ by it to obtain $u_1^{(k)}$.
4) Form $u^{(k)}$ via (4.12).
3. Let $u = u^{(r)}$ and form U^{-1}.
endalgorithm

It takes r stages to complete the algorithm, and at each stage it is necessary to perform the multiplication of a $(p2^{k-1})$th order Toeplitz matrix and a vector twice. In addition, at the first stage the pth order Toeplitz upper triangular matrix U_0 must be inverted. If all of these computations are performed by using the Toeplitz matrix-vector product algorithm described in Section 3, then the total time steps would be

$$S = (3\log^2 p + 5\log p) + (6\log p + 2) + \sum_{k=2}^{r}[8(k-1)2^{k-1} + 12\cdot 2^{k-1}\log p + 24(k-1)] = 4q(\log p + \log q) + 3\log^2 p + 12\log^2 q + 8q\log p - 12q - 13\log p - 12\log q + 18.$$

Thus, algorithm PITM is one with time complexity $O(q\log n + \log^2 p)$ and parallelism p. We wish to point out that the parallelism can easily be adjusted to match the hardware parallelism of the computer system to be used to perform the algorithm. Therefore, it is expected to gain high performance when the algorithm is used.

The algorithm may easily be used to solve Toeplitz triangular systems of the form

$$Ux = f, \tag{4.13}$$

the solution to which is given by

$$x = U^{-1}f. \tag{4.14}$$

So, we use Algorithm PITM to compute U^{-1} and then employ the algorithm in Table 1 to perform the multiplication in (4.14). The time complexity is the same as Algorithm PITM's. If $p << n$, then the complexity would become $O(q\log n)$. When $p = 1$, the sequential case, we have an $O(n\log n)$ algorithm.

References

[1] D. Bini, Parallel solution of certain Toeplitz linear systems, *SIAM J. Comput.*, **13** (1984), 268–276.
[2] Chen Ming-kui and Lu Hao, On the solution of certain Toeplitz systems, Proceedings of the First Conference on Parallel Algorithms in China, Beijing, 1987, 78–86.
[3] Chen Ming-kui, On the solution of circulant linear systems, *SIAM J. Numer. Anal.*, **24** (1987), 668–683.
[4] P. J. Davis, Circulant Matrices, John Wiley, New York, 1979.
[5] D. Hiller, A Survey of parallel algorithm in numerical linear algebra, *SIAM Review*, **20** (1978), 740–777.
[6] A.H. Sameh and R.P. Brent, Solving triangular systems on a parallel computer, *SIAM J. Numer. Anal.*, **14** (1977), 1101–1113.
[7] You Zhao-yong, Fast Algorithms in Linear Algebra and for Polynomial Computations, Shanghai Scientific and Technological Press, Shanghai, 1980. (in Chinese)